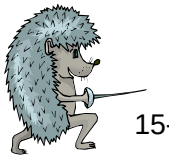
UNIVERSITY
OF OSLO

Sequence Diagrams - problems and methods

Version 050916

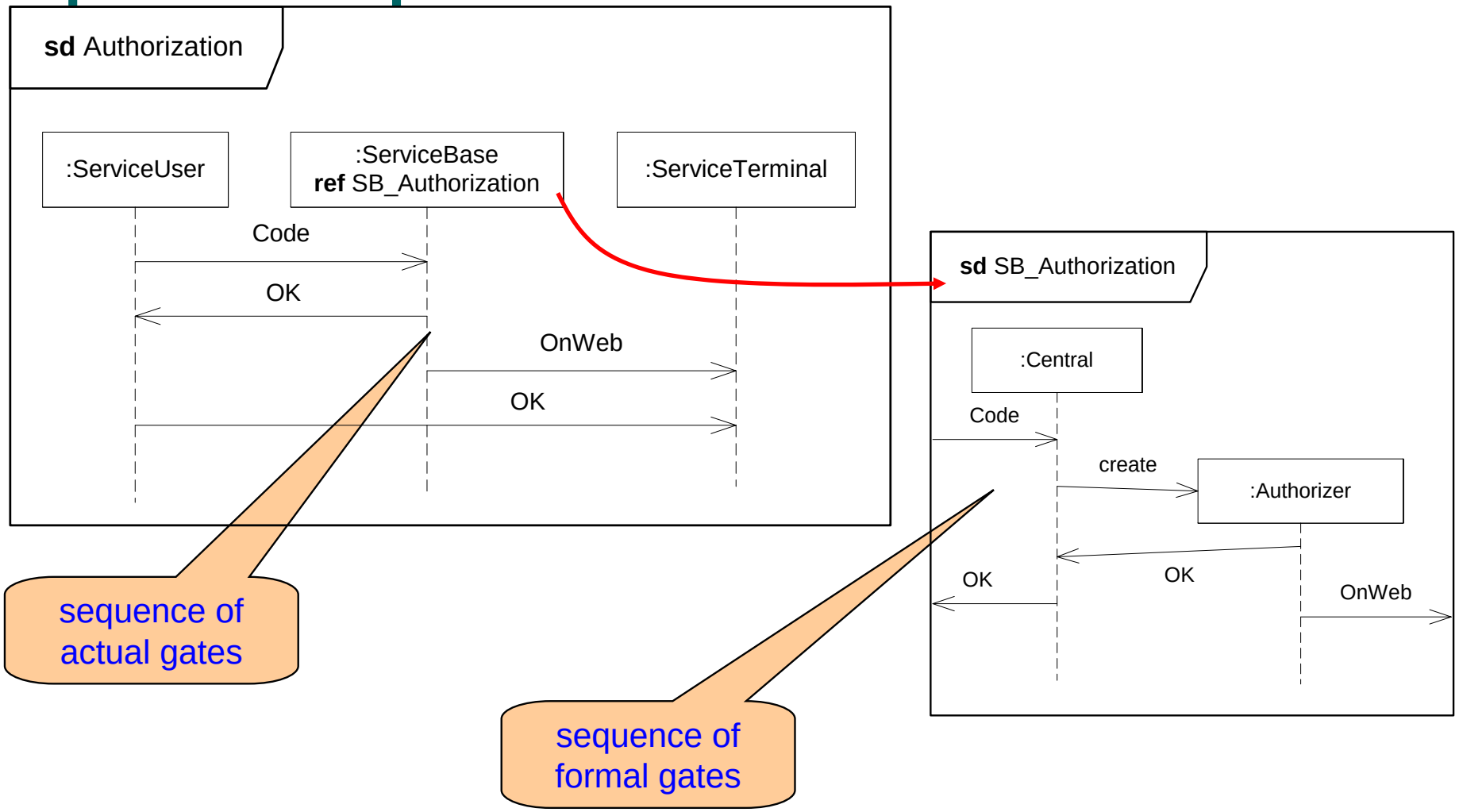
INF 5150



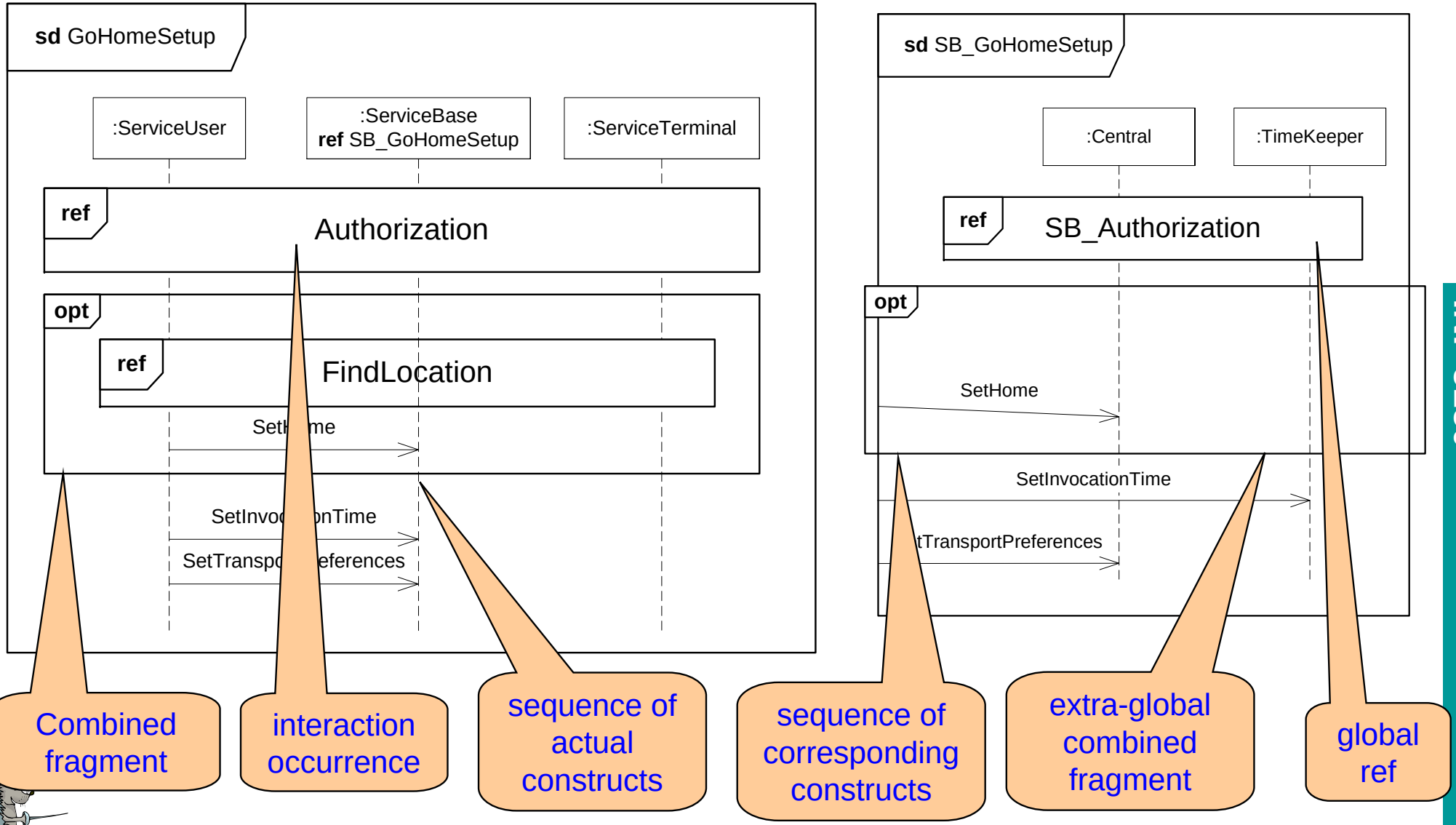
Problem areas

- Decomposition and References
  - how can we precisely define the combination of decomposition and references?
  - what about decomposition and combined interactions?
- Data
  - where is data in interactions?
  - what data can be involved in guards?

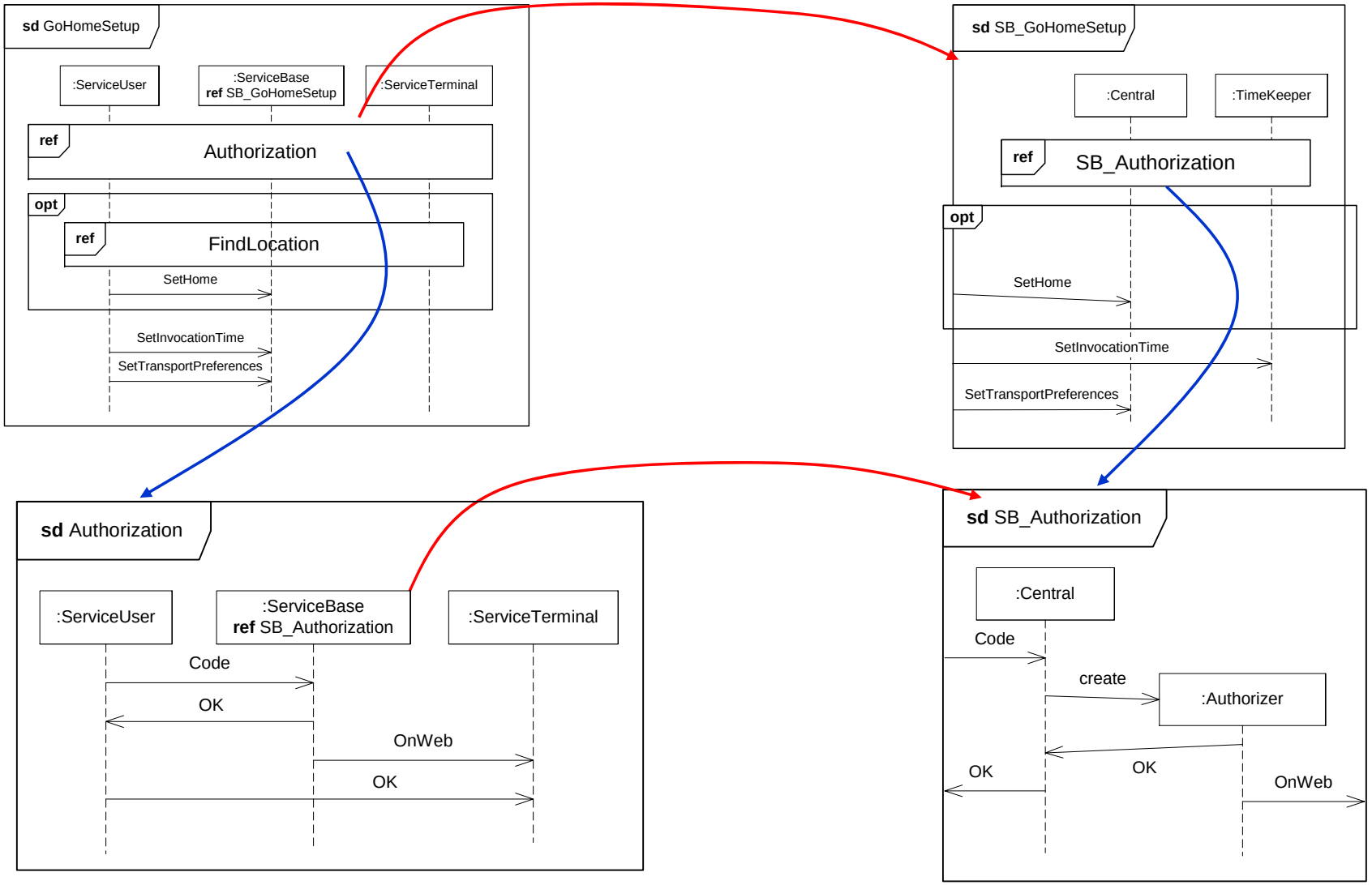
Simple Decomposition Revisited



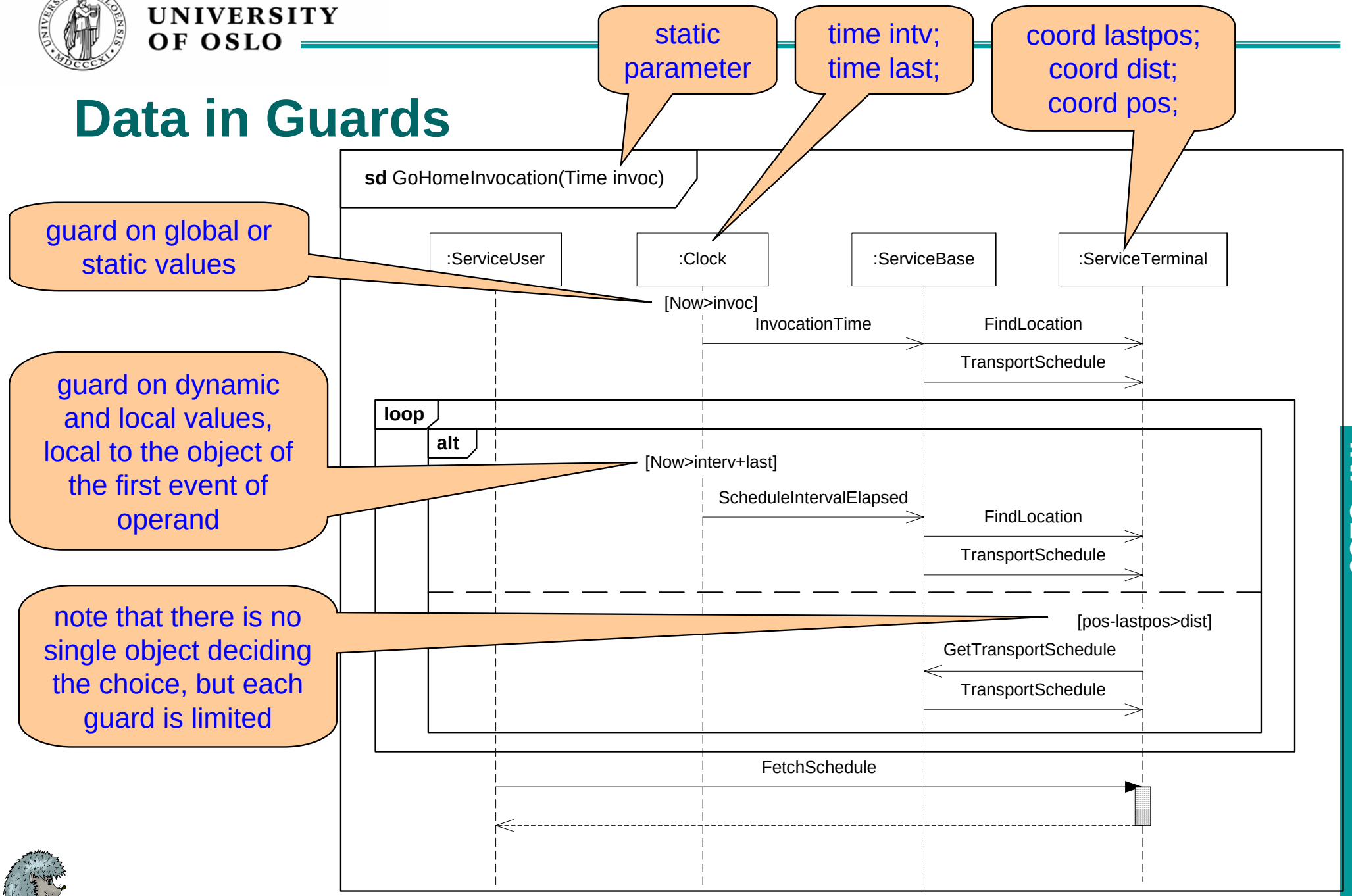
Decomposing covering ref's and combined fragments



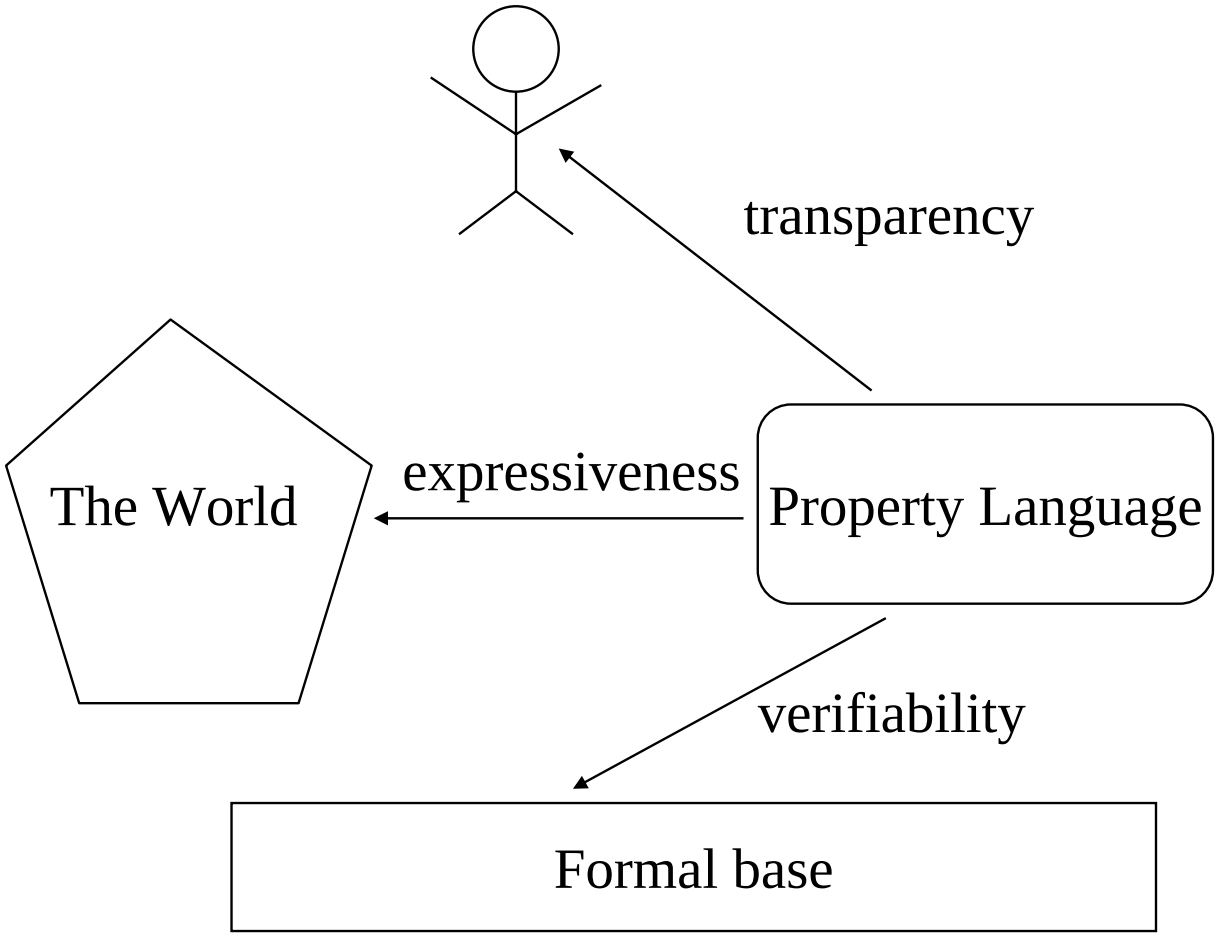
Commutative Decomposition



Data in Guards



Evaluating Property Languages



Comparison between Property Languages

Property	Prose	Seq. diag.	Math.	State Machines
transparency	good	good	poor	good
expressiveness	fair	fair	good	good
formalization	poor	good	good	good
liveness, safety	fair	fair	good	fair
overview	good	good	fair	good
interaction	poor	good	fair	fair
time req.	fair	fair	good	poor
capacity	fair	poor	fair	poor

Basic Sequence Diagram Methodology

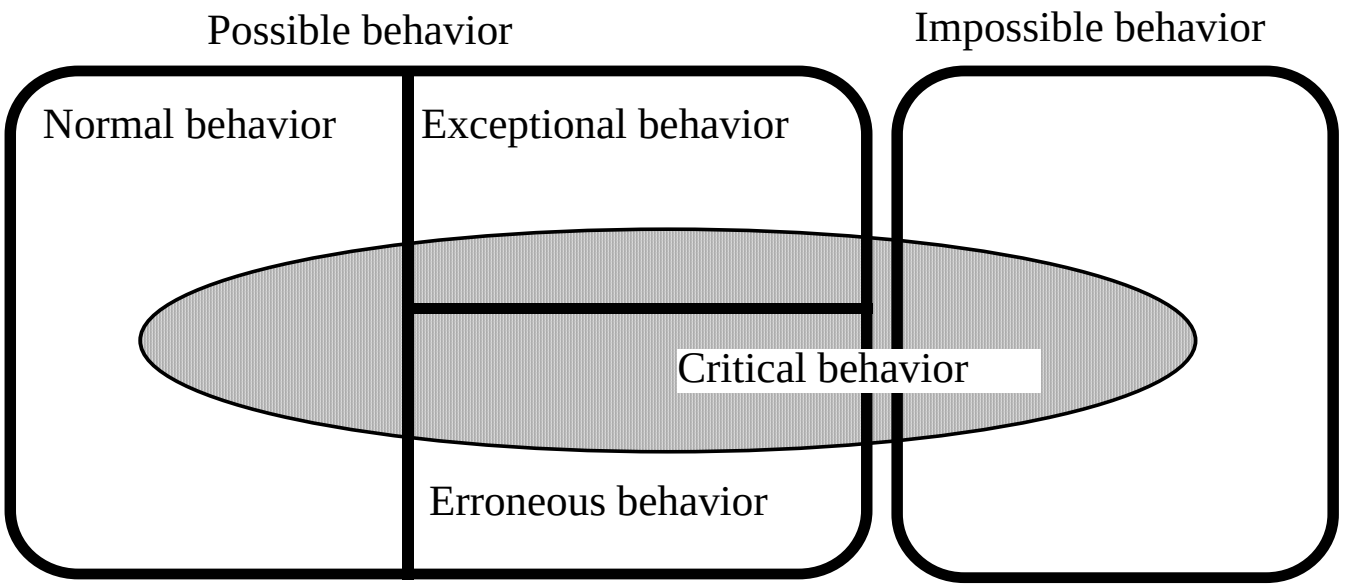
Even though Sequence Diagrams are simple and may be read and produced by engineers without much formal training, it is possible to:

- make beautiful diagrams that say nothing,
- make messy diagrams that are meant to convey critical information,
- make terrible diagrams in an early phase that make it impossible to design a sensible system in a later phase.
- use extensions to UML/MSC that are not standard and that may prevent you from using (more than one) tools.

The methodology aims at bridging the gap between the notation and the development process using it.

Seq. diag. classification: case evaluation

- *Normal* behavior is the behavior that we expect
- *Exceptional* cases are those that may happen, and that we should prepare for, but which we do not consider normal.
- The *erroneous* behavior is behavior that we try to avoid, but which should not destroy our system.
- *Impossible* behavior is behavior that cannot happen



Seq. diag. classification: descriptive goal

Descriptive goal	Target audience	Life span
historical	project members, managers, potential customers	temporary
documentary	managers, customers	negotiations or product span
requirements	customers, project team	product span
design	project team	project
test	testers, customers	product span

Step 0: Make explicit the company SD strategy

- *Tools*: What tools will be used to produce and maintain the mscs?
- *Coverage Profile*: How do the diagrams cover the universe of scenarios?
- *Document Profile*: What diagrams are to be produced?
- *The Inexpressible*: How is information not expressible in UML/MSC attached?

Step 1a: the first sequence diagrams

- Our metaphor for building our MSC document is a news photographer covering a major event.
- Firstly he will make sure to take pictures of the main characters – the *normal* cases.
- Then he will look for some *exceptional* situation which might sell better to the public and which may capture unexpected problems like the police horse galloping.
- Then he digs for *errors* like the possible assassin in the bushes.
- Finally he could illustrate the *impossible* by manipulating a picture like placing Forrest Gump with President Nixon.

Step 1b: Establish the interplay with non-developers

- Require responsibility and approval from the non-developers;
- Involve the non-developers in making additional diagrams making sure that they understand UML/MSC and that they understand that they understand UML/MSC;
- Associate concrete input/output with the user interface.
- Encourage the non-developers to use their UML/MSC knowledge during the design and model checking phases

Summary Basic Sequence Diagram strategy

Step 0 : company strategy
- what tools
- what coverage
- what MSC documents
- How to attach informal text

Step 1a : the first mscs
- normal
- exceptional
- erroneous
- impossible
- critical

Step 1b : interplay with non-developers
- require responsibility
- active involvement
- be concrete
- encourage further use of MSC

Step 2a : Variants and similarity
- global conditions
- road map
- MSC document table

Step 2b : Refinement
- message hierarchy
- instance hierarchy

Step 2c : Inexpressibles
- dependency
- capacity and duration

Step 3 : Support the design
- alignment table
- checking existence
- checking full coverage

Step 4 : Test mscs
- isolate IUT
- project existing mscs

STAIRS - Steps To Analyze Interactions with Refinement Semantics

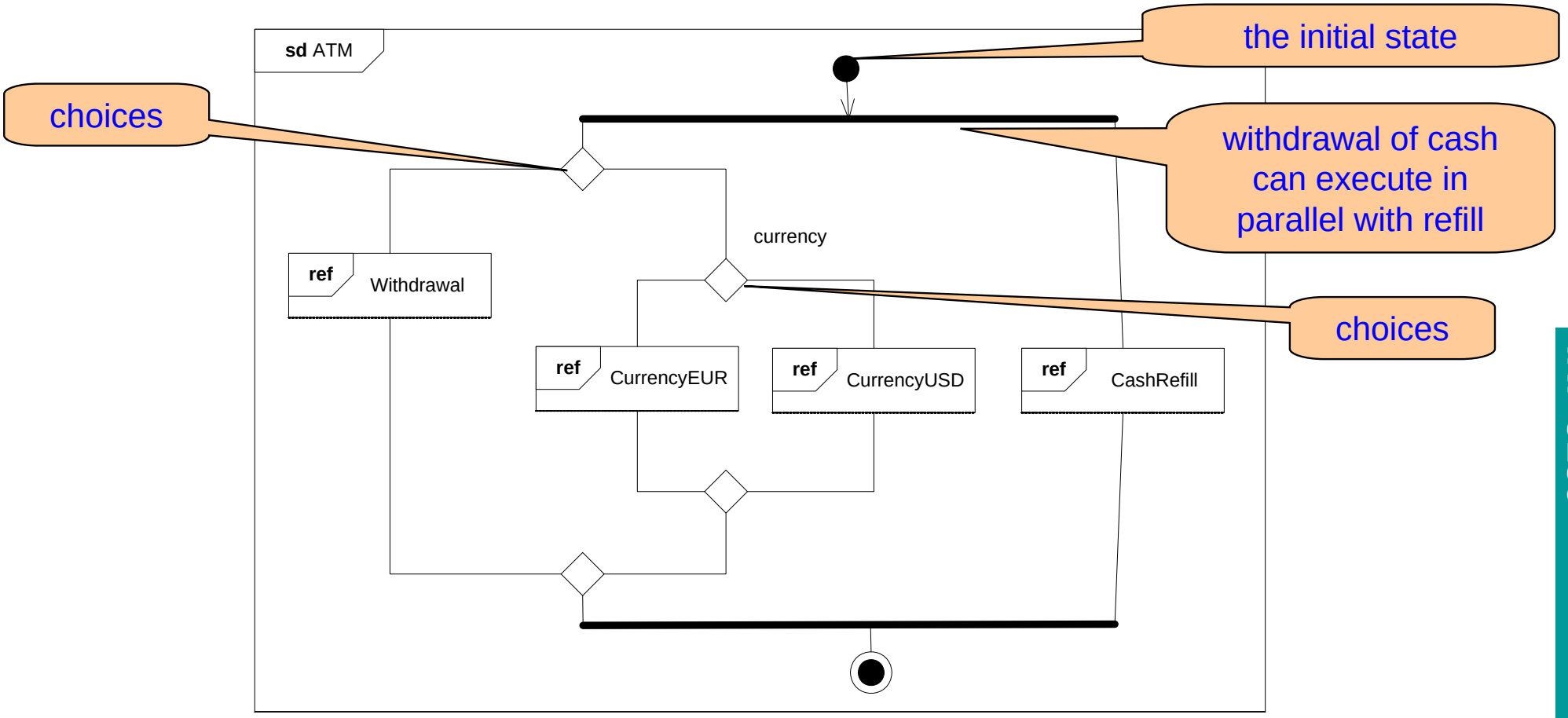
supplementing

£ ¥ € $

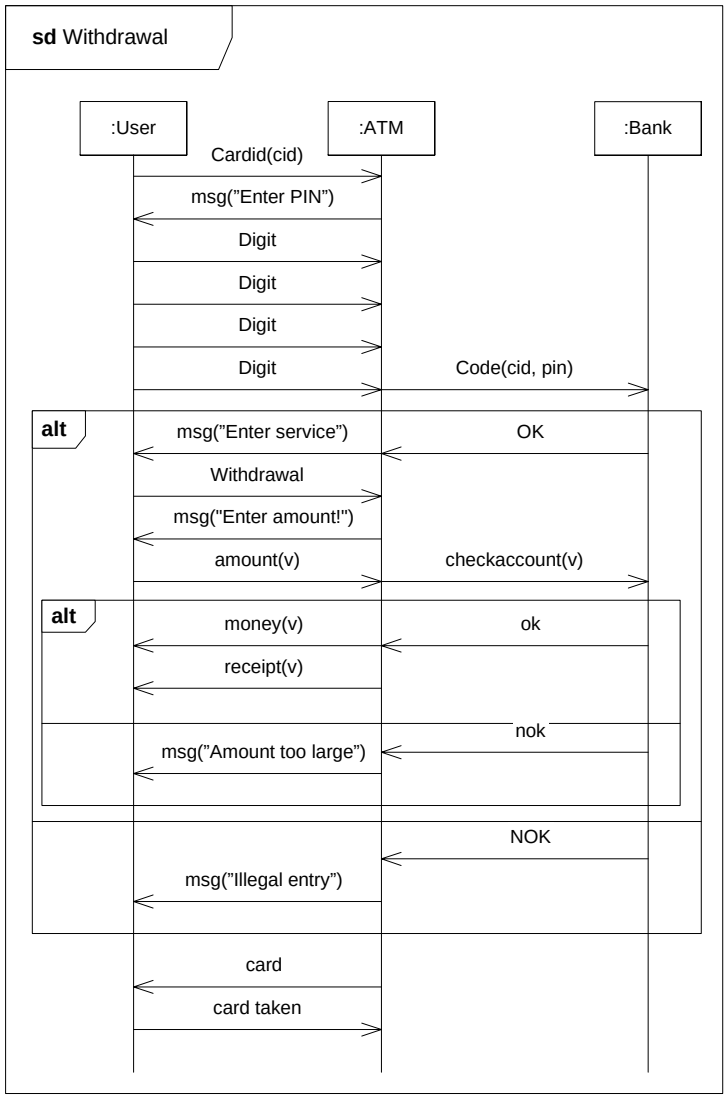
narrowing

detailing

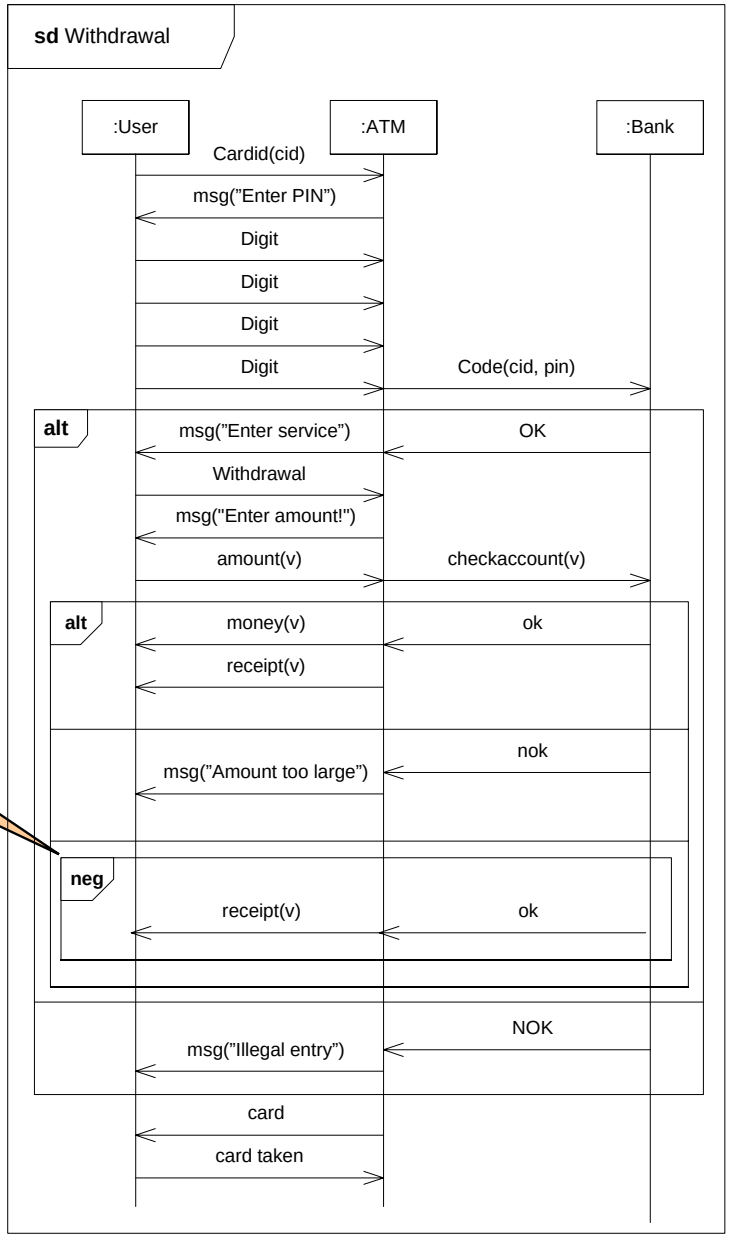
Automatic Teller Machine



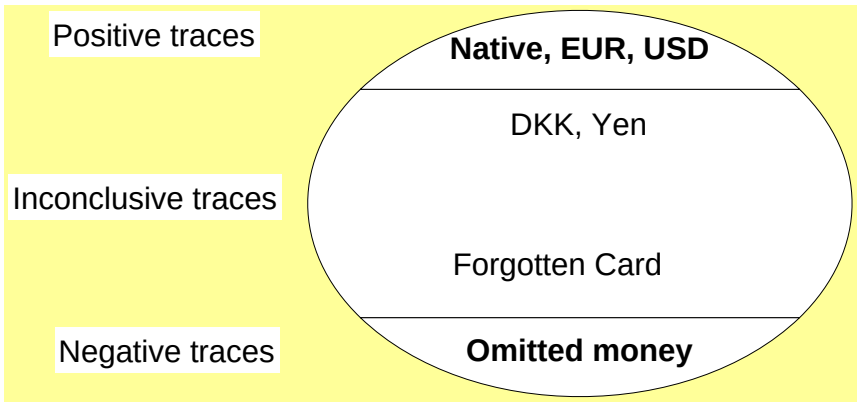
Some positive traces of Withdrawal

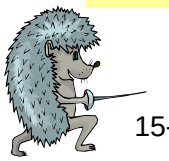


Negative experiences

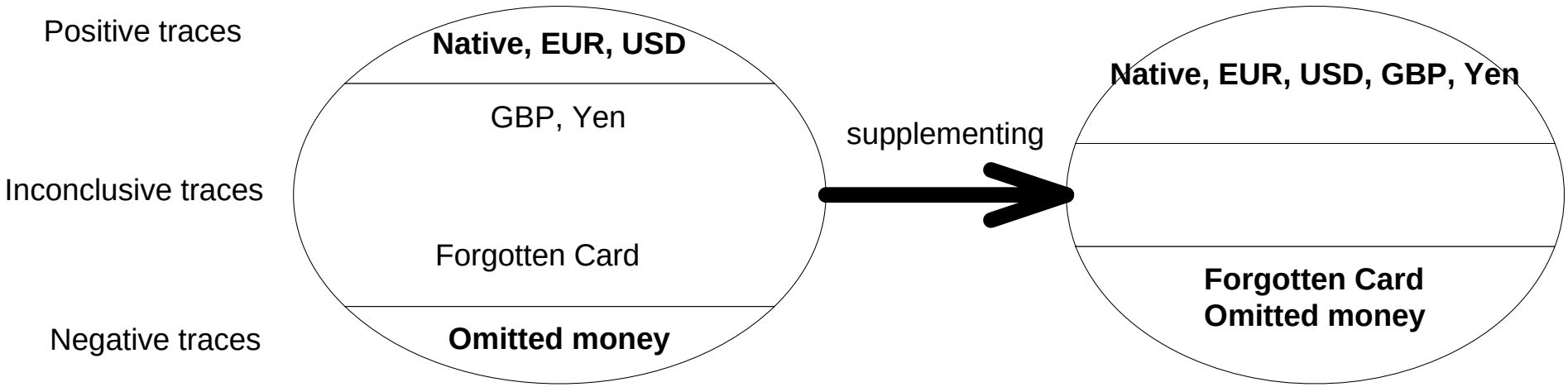


negative traces

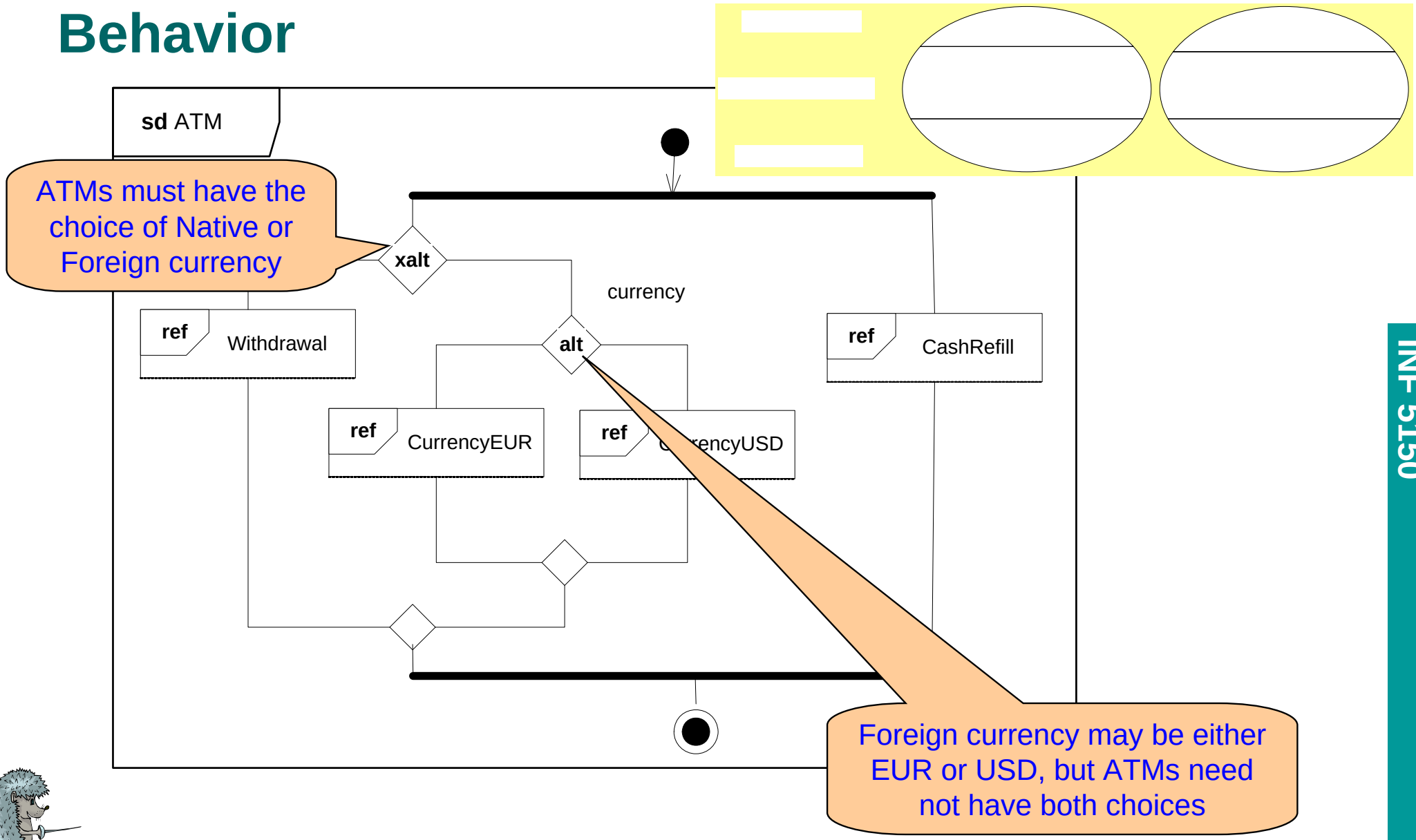




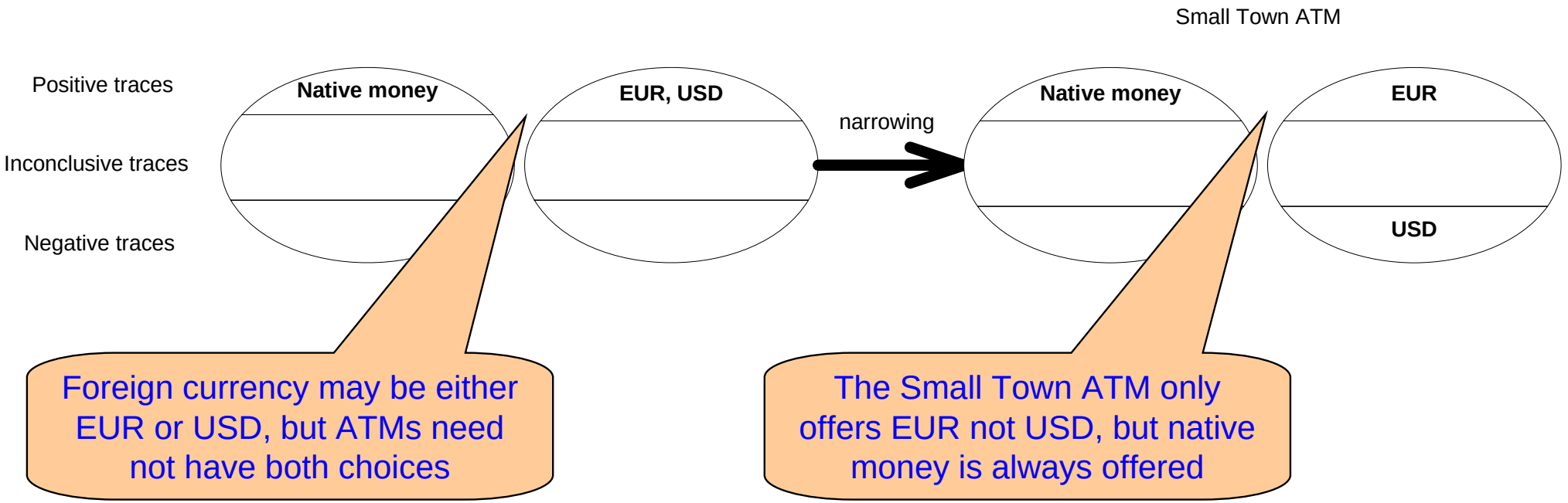
Supplementing – more potential traces



Distinguishing Mandatory from Potential Behavior

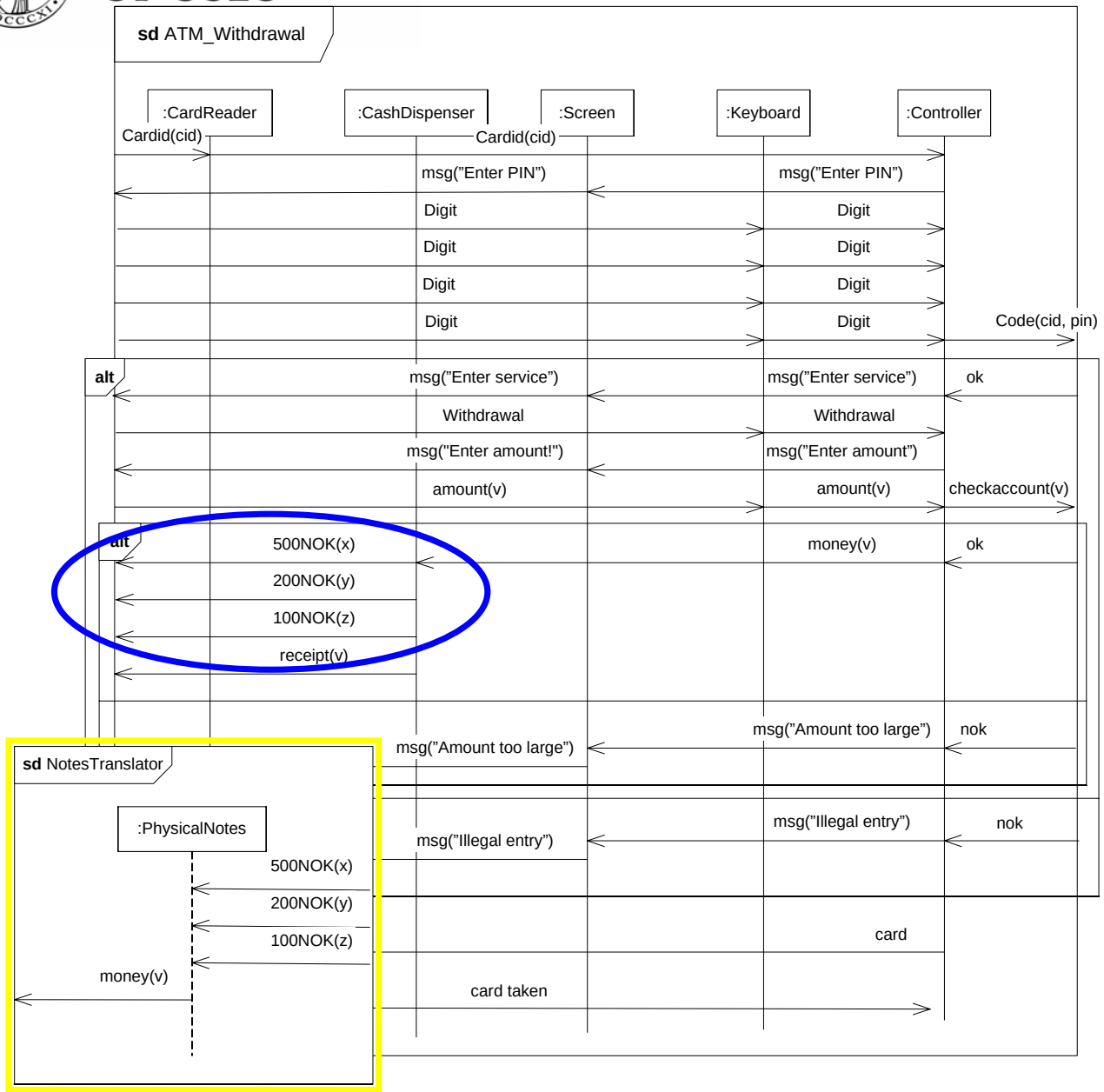


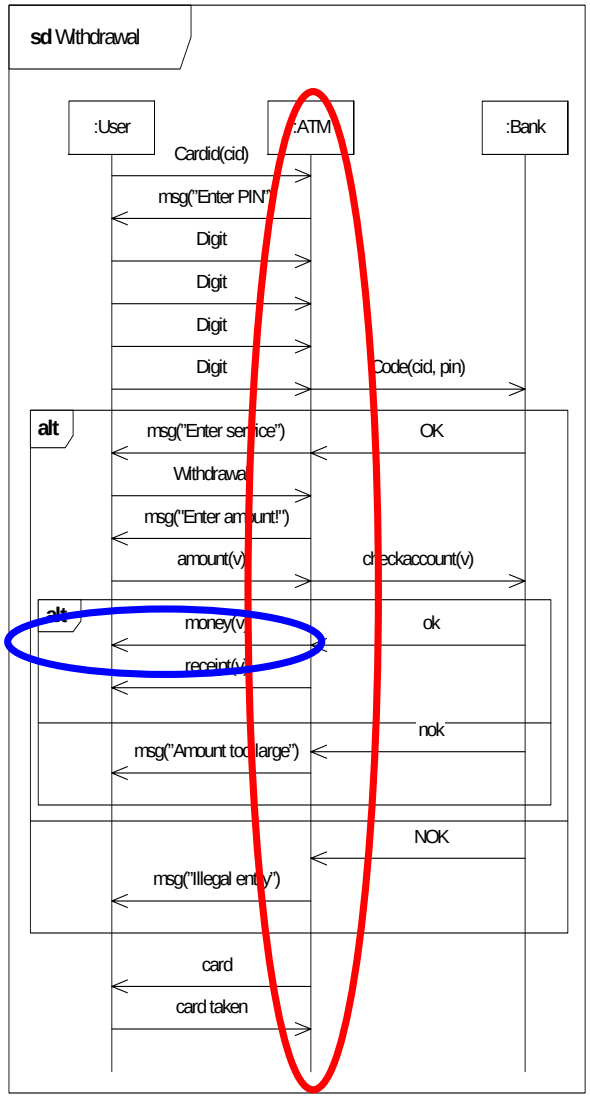
Narrowing to specify your implemented ATM



UNIVERSITY OF OSLO

Detailing





STAIRS for Dolly

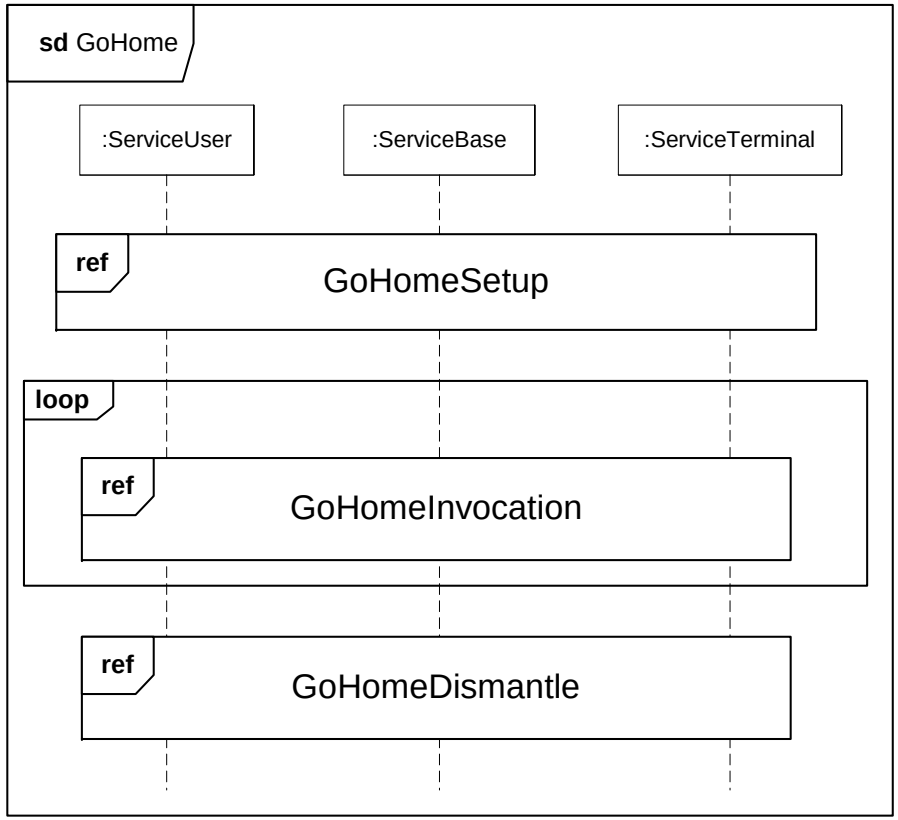
- Make more concise
  - by putting the specification in a more formal language
- Refine
  - supplement
    - add new aspects/runs
  - narrow
    - reduce underspecification
  - detail
    - divide and conquer
      - decomposition
      - breaking down messages
      - reveal new significant objects and behavior
- Distill
  - summarize and clean up

Service: Dolly Goes To Town

- Informal specification:
  - ”Dolly gets transport schedules when she wants to return home from town. The schedules take her from where she is to her home”

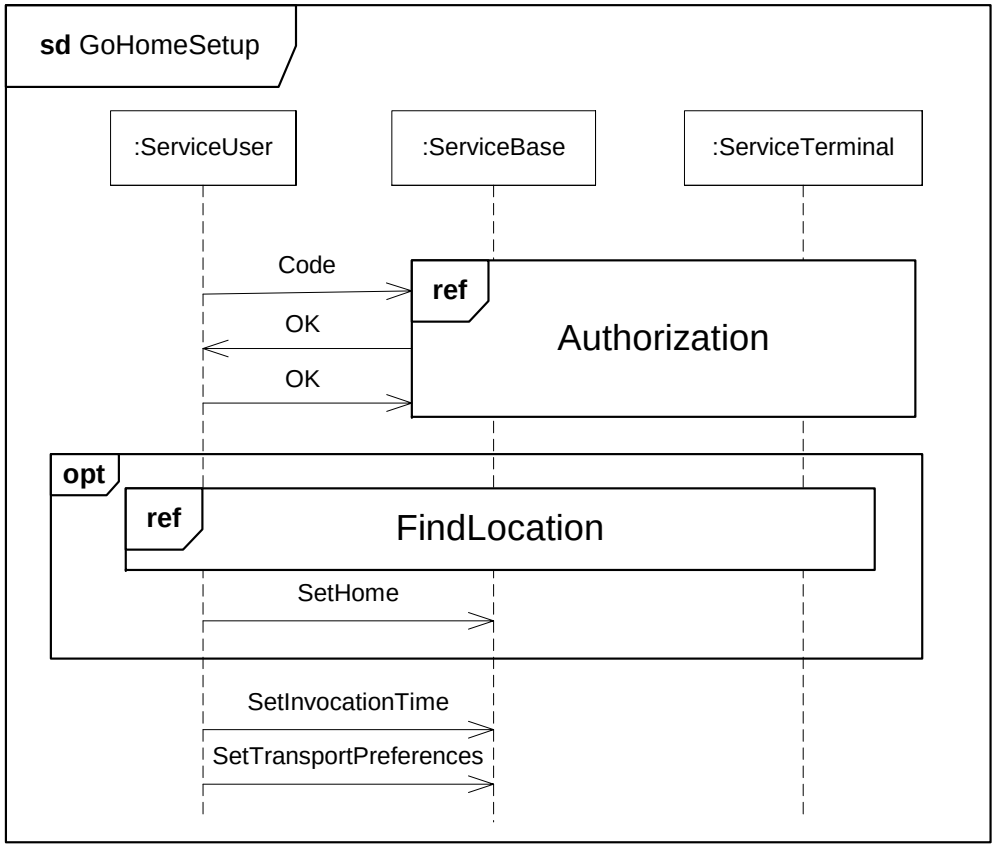
Make more concise – formalize

- ”Dolly gets transport schedules when she wants to return home from town. The schedules take her from where she is to her home”



Refine – supplement

- ”Dolly gets transport schedules when she wants to return home from town. The schedules take her from where she is to her home”
  - what happens if she is not authorized during setup?
  - what if her home location is not set, or it is wrong?

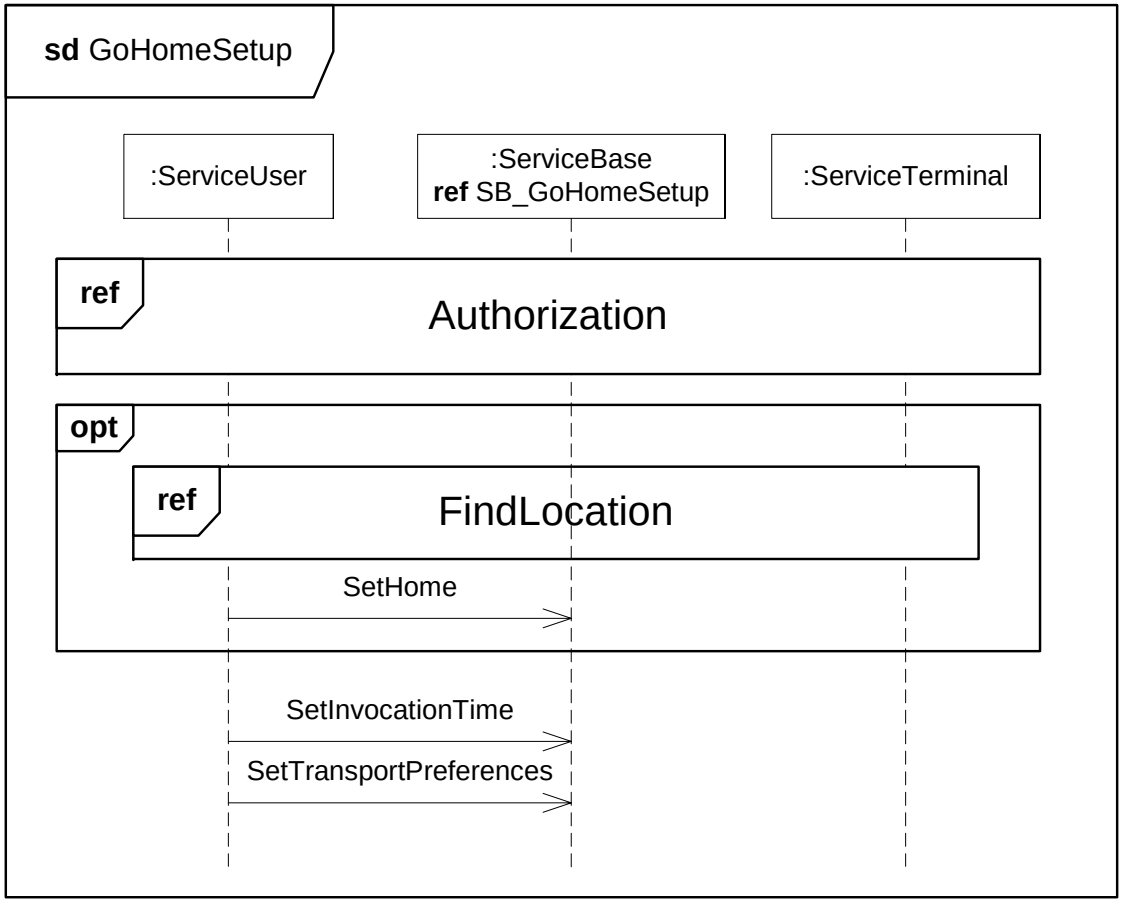


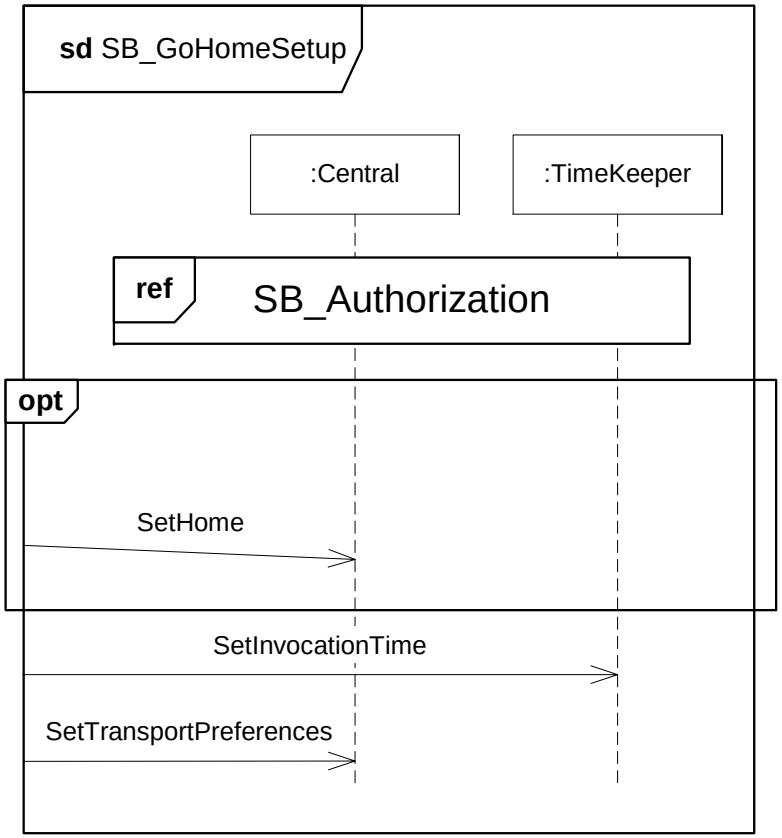
Refine – narrow

- ”Dolly gets transport schedules when she wants to return home from town. The schedules take her from where she is to her home”
  - Dolly subscribes to a schedule service from her web-browser
  - Dolly gets the schedules on her mobile terminal
  - Dolly gets the schedules via SMS
- Is the formalization consistent with the narrowing?
  - Perhaps! The two descriptions are not obviously in harmony, but rather focus on different aspects

Detail – *decompose*

- decomposing the ServiceBase



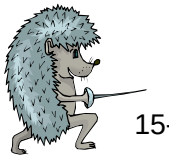


Detail – reveal

- ”Dolly gets transport schedules when she wants to return home from town. The schedules take her from where she is to her home”
  - The presentation is not significant on the domain level, but becomes visible on the design level
    - how shall the schedule be presented?
    - should the different schedules be sorted in a particular way?
    - how should the information retrieved be destroyed?

Distilling

- During the process from informal understanding to more precise and more detailed description
  - the number of diagrams have grown
  - changes have been made in some diagrams that should have implied changes in other diagrams
    - (not saying that anything is actually wrong, but not absolutely uniform)
  - new understanding has been reached that has not really had impact on all corners of the design
- Distilling means
  - to purify the description
    - to show uniform approaches to problems
    - that similar situations appear similar (and vice versa)
    - that good overview is achieved and the new understanding properly conveyed to others



How is this related to unassailability?

- UML 2 sequence diagrams are
  - intuitive
    - but only partial
  - precise
  - supported by tools
- Proper methodology is needed
  - recognizing that sequence diagrams do not tell the whole story
  - increasing the consciousness of
    - which diagrams to make
    - their purpose
- Achieving
  - early awareness of problems